Exceptions Lab

Pre-requisites: C++ Interlude 3 – Exceptions

For this lab you'll begin with the following starter files:

- Account.cpp – This file contains the Account class which stores a bank balance. It includes:
    - `double deposit(double amount)` which deposits amount into the account and returns the new balance or returns a -1 if amount is less than zero.
    - `double withdraw(double amount)` which deducts amount from the balance and returns the new balance if there are sufficient funds, otherwise it returns a -1 for insufficient funds.
- main.cpp – A basic driver for testing.

Returning -1 as an indicator of an error works, but using exceptions is a better option. Your task will be to update `deposit()` and `withdraw()` to use exceptions.

Step 1

Create a new Visual Studio project and add both Account.ccp and main.cpp to it.

Step 2

At the `// Step 2` comment in Account.cpp, add a programmer-defined exception class called `NegativeAmount` which inherits from the `logic_error` exception class. Your programmer-defined exception needs a constructor that receives a string parameter and passes its value to the `logic_error` constructor.

See page 238 of the textbook if you need help with the syntax.

Step 3

Add a programmer-defined exception class called `InsufficientFunds` to Account.cpp.

Step 4

Update `deposit()` to throw a `NegativeAmount` error when `amount` is less than zero rather than returning -1. Throw the exception with the message "Error, you can't deposit a negative amount!"

Step 5

Update `withdraw()` to throw a `NegativeAmount` exception with the message "Error, you can't withdraw a negative amount!" when `amount` is less than zero rather than returning -1.

Also have it throw an `InsufficientFunds` exception with the message "Error, account has insufficient funds for withdrawl!" when `amount` > `balance` rather than returning -1.

Step 6

main.cpp creates an Account object called `a` with a balance of $100. If prompts the user to choose which exception to test and how.

In main.cpp, add a try block which:

- Prints the starting balance of `a`.
- Calls `a.deposit(-200)` if `choice` == 1
- Calls `a.withdraw(200)` if `choice` == 2
- Calls `a.withdraw(-200)` if `choice` == 3

After the try block, add 2 catch blocks, one for `NegativeAmount` and one for `InsufficientFunds`. Each catch should:

- Print a message that the exception was thrown.
- Call the exception's `what()` method to print the exception's message.

Step 7

Make sure to test all 3 menu options and verify the output. When you're finished it should look something like this:

```
What test do you want to do?
1. NegativeAmount exception with deposit()
2. InsufficientFunds exception with withdraw()
3. NegativeAmount exception with withdraw()
Your choice: 1

Working with an account that has a $100 balance.
Trying a deposit of -$200.

***NegativeAmount exception was thrown.***
Here is the exception message:
Error, you can't deposit a negative amount!
```

Turn In

Place your name in a comment at the top of both starter files, zip them together (do not include the Visual Studio project files), and upload them before the due date.